Section 6 Concepts

Real-world data is often semistructured. Querying semistructured data presents additional challenges, compared to (structured) relational data. For example, semistructured data may have missing or additional attributes, heterogeneous types, and nested structure.

Nevertheless, it can be convenient to write queries on semistructured data in its original form, i.e., without transforming or otherwise preprocessing the data into a "cleaner" form. AsterixDB is one DBMS that facilitates the analysis of semistructured data.

An alternative is to transform semistructured data into a relational form, and then run relational queries on the data in its new form. The initial transformation can require effort, but the ease gained in writing queries may be worth it in the long term. Semistructured data and relational data are less different than you might imagine. We encourage you to practice transforming semistructured data and queries into a relational form on your own.

AsterixDB Terms

A *dataverse*—short for “data universe”—is a place (similar to a database in a relational DBMS) in which to create and manage the types, datasets, functions, and other artifacts for a given AsterixDB application. A special dataverse called *Metadata* contains AsterixDB's metadata.

A *type* is like a relational schema, but much looser (allows additional attributes if open, flexible type, allows missing values, nested data, etc.).

A *dataset* is like a relational table. It contains the data itself as a collection, as well as indexes for accessing the data. There is always at least one index called the primary index, specified at creation time.

A *collection* is either an array (ordered) or multiset (ordered). Both allow duplicates.

An *object* is like a relational tuple. It contains *attributes* known as name-value pairs (in the JSON lingo) or key-value pairs (in the ADM lingo). The ADM lingo is more accurate because it emphasizes the fact that keys (attributes names) are unique within an object.

Worksheet

Use the Mondial dataset in hw5 to solve the following problems.

1. Return the set of all mountains, i.e., as a single tuples with an attribute containing the collection of all mountains.

2. Return each mountain one by one, i.e., as a collection of tuples that each contain a single mountain. Compare it to Problem 1.

3. Return name and type for each mountain, in descending order of the height.

4. Find mountains located in more than 1 country. Your query should return mountain name and the count.

5. For each country, return the country name and a list of all the mountain names in that country.

Suppose that we store all the data for our social network in a single dataset of Users:

[{“handle”: “biebs”,

“name”: “Justin Bieber”, “home_city”: “Somewhere, Canada”, “bio”: “...”, “friends”: [“kimkardashian”, “shaq”, ...], “messages”: [{“text”: “:-* :-* :-* :-*”, “from_city”: “Los Angeles, CA”}, {“text”: “New. Music. Friday.”, “from_city”: “Los Angeles, CA”},

...] } ...]

1. For each home city, compute a list of users from that home city. Your query should return a list where each element consists of a home city name and a list of user handles.

2. Return pairs of users that have at least one common friend.